Dramatic Improvements to Feature Based Stereo

V.N. Smelyansky, R.D. Morris, F.O. Kuehnel, D.A. Maluf, and P. Cheeseman

NASA Ames Research Center, MS 269-2, Moffett Field, CA 94035, USA
[vadim,rdm,kuehnel,maluf,cheesem]@email.arc.nasa.gov

Abstract. The camera registration extracted from feature based stereo is usually considered sufficient to accurately localize the 3D points. However, for natural scenes the feature localization is not as precise as in man-made environments. This results in small camera registration errors. We show that even very small registration errors result in large errors in dense surface reconstruction.
We describe a method for registering *entire* images to the inaccurate surface model. This gives small, but crucially important improvements to the camera parameters. The new registration gives dramatically better dense surface reconstruction.

1 Introduction

The goal of surface recovery is to take a set of images and estimate the positions and orientations of the cameras that produced the images, and a representation of the surface that was imaged. This is an example of an *inverse problem.* The forward (or direct) problem is: given a surface and the position and orientation of a camera, what is the expected image? This is the area of computer graphics known as rendering [1]. The inverse problem is: given a set of images, estimate the position and orientation of the cameras, and the shape and reflectance properties of the surface. That is, estimate a *generative model* [2, 3].

The conventional feature based approach to 3D surface reconstruction takes a sparse set of corresponding feature points from which the positions and orientations of the cameras are estimated. The quality of the camera calibration crucially depends on well localized features. Feature tracking in a sequence of images with small frame to frame disparity has been demonstrated successfully. The two main concerns are the robustness and the accuracy of such an approach. Robustness is usually improved by tracking across a sequence with small inter-frame displacements, but for many applications this cannot be assured. A further concern is that the overall accuracy of the reconstructed 3D model from a sparse point could is rather doubtful and prior knowledge is not easily incorporated in the conventional reconstruction scheme.

We show that a robust and accurate reconstruction scheme that can incorporate any prior knowledge can be implemented by applying Bayesian inference of the underlying model space. We postulate *models* for the surface and for the imaging process, and Bayes theorem tell us how to estimate the parameters of these models from the image data. We show that this approach allows us to make

small but crucially important improvements to the camera parameters estimated from point matching. These improvements result in a *dramatic improvement* in the accuracy of the 3D surface model.

In this paper we restrict our reconstruction to simple surface models (no occluding parts), therefore we use a simple triangulated mesh model for the geometry of the surface, storing heights, z, at each vertex of the mesh. We also associate a parameterized reflectance model with the surface. For simplicity here we consider the Lambertian model, and store a single albedo value, ρ at each vertex. (For multispectral data we store an array of albedo values, one for each spectral band.)

We use the standard pinhole camera model for the image formation process [9], and assume that the internal camera parameters are known. (See, for example, [10] for a simple method of internal camera calibration.) The theoretical development of our approach can be generalized to other imaging geometries and surface reflectance models.

The closest work to that described here is in [4, 3]. That work also used a triangulated mesh as the surface representation. The cost function they used is based on minimising the variance of the grey levels of the vertices' projection into the images, rather than the direct image error that is used here. The approach in [4, 3] is thus restricted to triangulated meshes that are coarse when projected into the images. The approach described here places no restrictions on the density of the mesh, which may be super-resolved [5]. The system in [4, 3] is also restricted to cases wher the lighting was from the same direction in all images. Here we require only that the lighting direction is known.

Thus we wish to infer the heights, $\mathbf{z}$, the albedos, ρ and the camera parameters, Θ, from the images. Bayes theorem gives

$$p(\mathbf{z}, \rho, \Theta | \{I\}) \propto p(\{I\} | \mathbf{z}, \rho, \Theta) p(\mathbf{z}, \rho, \Theta) \tag{1}$$

We assume that the priors are independent, so that

$$p(\mathbf{z}, \rho, \Theta) = p(\mathbf{z}) p(\rho) p(\Theta)$$

and currently we use a simple smoothness prior for $\mathbf{z}$ and ρ based on penalizing curvature, and a uniform prior on Θ. These initial prior assumptions are made for the sake of simplicity, and are not fundamental to the approach (see below). The likelihood is assumed to result from Gaussian errors between the image $\hat{I}(\mathbf{z}, \rho, \Theta)$ synthesized from the surface model and the observed images $\{I\}$, giving

$$p(\{I\} | \mathbf{z}, \rho, \Theta) \propto \exp\left(\sum_{f,p} \left(I_{f,p} - \hat{I}_{f,p}(\mathbf{z}, \rho, \Theta_f) \right)^2 / (2\sigma_e^2) \right) \tag{2}$$

where the sum is over all pixels, p in all images, I_f. The surface parameters, $\mathbf{z}$, ρ, are clearly shared between all images. Each image has its own set of camera parameters, Θ_f.

The function $\hat{I}(\mathbf{z}, \rho, \Theta)$ is the process of *rendering* the surface described by $\{\mathbf{z}, \rho\}$ with the camera location and orientation given by Θ. This is clearly non-linear, and makes optimization of the posterior distribution in equation 1 difficult. To make progress in finding the maximum a-posteriori (MAP) estimate, we linearize the image formation process about the current estimate,

$$\hat{I}(\mathbf{z}, \rho, \Theta) = \hat{I}(\mathbf{u}_0) + \mathbf{D}\mathbf{x} \tag{3}$$

where $\mathbf{u} = \{\mathbf{z}, \rho, \Theta\}$, $\mathbf{x} = \mathbf{u} - \mathbf{u}_0$ and

$$\mathbf{D} = \left\{ \frac{\partial \hat{I}}{\partial \mathbf{z}}, \frac{\partial \hat{I}}{\partial \rho}, \frac{\partial \hat{I}}{\partial \Theta} \right\}$$

If we use a Gaussian smoothness prior with covariance matrix Σ as described above then the linearization converts finding the MAP estimate to the minimization of a quadratic form

$$L = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}\mathbf{x} \tag{4}$$

$$\mathbf{A} = \Sigma^{-1} + \frac{1}{\sigma_e^2}\mathbf{D}\mathbf{D}^T \tag{5}$$

$$\mathbf{b} = \frac{I - \hat{I}(\mathbf{z}, \rho, \Theta)}{\sigma_e^2}\mathbf{D} \tag{6}$$

which is equivalent to the solution of the system of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{7}$$

Note that for this linearization to proceed, the only restriction on the smoothness prior is that it can be expressed as a covariance matrix. This makes no assumption of spatial uniformity; indeed the prior can easily be made spatially adaptive, to allow for the formation of discontinuities in the heights and albedos.

Consider the structure of this system of equations. The matrix of derivatives $\mathbf{D}$ is of dimensions

$$\begin{aligned}(\text{no. of pixels}) \times (\text{no. of heights} + \text{no. of albedos} + \\ \text{no. of camera parameters})\end{aligned} \tag{8}$$

or, for the results presented later

$$(256 \times 256) \times (301 \times 301 + 301 \times 301 + 6)$$

The portion of this matrix that is due to the differentials with respect to $\mathbf{z}$ and ρ is very sparse, as typically each mesh vertex is used by a few of the triangles that make up the surface, and these triangles project into only a few pixels. The portion due to the differentials with respect to the camera parameters is, however, dense, as changing any one of the camera parameters typically affects

the intensities of all the pixels in the image. As a result of this, $\mathbf{D}\mathbf{D}^T$ and hence $\mathbf{A}$ are very large (around $(180,000 \times 180,000)$ and *dense* (around 3×10^{10} elements). It is clearly impractical to perform joint estimation in this manner. Instead we estimate alternately the camera parameters and the surface parameters, that is

$$\begin{aligned} &\text{given } \Theta, \text{ estimate } \{\mathbf{z}, \rho\} \\ &\text{given } \{\mathbf{z}, \rho\}, \text{ estimate } \Theta \end{aligned} \tag{9}$$

In this way we compute either with a very large, but very sparse matrix when estimating $\mathbf{z}$ and ρ, or with a very small, dense matrix when estimating Θ. The estimates are made by using conjugate gradient to solve equation 7 in an iterative manner. At convergence, we update the current estimate, $\mathbf{u}_1 = \mathbf{u}_0 + \mathbf{x}$, re-render to compute new values of $\hat{I}(\mathbf{z}, \rho, \Theta)$ and $\mathbf{D}$, and repeat the solution of equation 7 until a stable solution is reached. This optimization process can be applied in a multiresolution framework, to both accelerate and improve convergence.

This requires an initialization for either Θ or $\{\mathbf{z}, \rho\}$. We use initial values for Θ from point matching a *very small* number of points, or from nominal camera position and orientations, if they are known (eg from rover or aircraft dead-reckoning). In the experiments described later, point matching was used.

The remainder of the paper is organized as follows: In section 2 and 3 we describe the basic rendering algorithm for a renderer which efficiently computes the images *and* calculates the derivative values used for the conjugate gradient search outlined in section 1. Results and Conclusions are given in sections 4 and 5

2 The Fractional Derivative Renderer

As we have seen, to solve the inverse problem we must be able to simulate the forward problem, to compute $\hat{I}(\mathbf{z}, \rho, \Theta)$, ("rendering"). Current rendering technology uses "image space" computation, where the fundamental unit is the pixel. Each pixel is assumed to be illuminated by light from one, and only one, triangular facet. This assumption makes for very fast rendering, but results in aliasing artefacts. It also makes the rendering process non-differentiable.

To enable a renderer to also compute derivatives it is necessary that all computations are done in "object space". This implies that the light from a surface triangle, as it is projected into a pixel, contributes to the brightness of that pixel with a weight proportional to the fraction of the area of the triangle which projects into that pixel. The total brightness of the pixel is thus the sum of the contributions from all the triangles whose projections overlaps with the pixel

$$\hat{I}_p = \sum_{\Delta} f_{\Delta}^p \, \Phi_{\Delta}, \tag{10}$$

where f_{Δ}^p is the fraction of the flux from triangle Δ that falls into pixel p, given by

$$f_{\Delta}^p = \frac{\bar{A}_{\text{polygon}}}{\bar{A}_{\Delta}}, \tag{11}$$

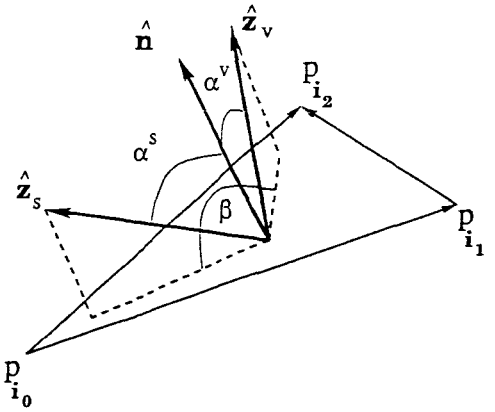

Fig. 1. Geometry of the triangular facet, illumination direction and viewing direction. $\hat{\mathbf{z}}_s$ is the vector to the illumination source; $\hat{\mathbf{z}}_v$ is the viewing direction.

where $\bar{A}$ denotes projected area, and $\Phi_\triangle$ is the total flux from the triangle, and $\bar{a}_{\text{polygon}}$ is the area on the image plane of the intersection of the projection of the triangle and the pixel. In the case of Lambertian reflection, this is given by

$$\begin{aligned} \Phi_\triangle &= \rho E(\alpha^s) \cos\alpha^v (\cos\theta)^\kappa \Delta\Omega, \\ E(\alpha^s) &= \mathcal{A}\left(\mathcal{I}^s \cos\alpha^s + \mathcal{I}^a\right). \\ \Delta\Omega &= S/d^2. \end{aligned} \tag{12}$$

Here ρ is an average albedo of the triangular facet. Orientation angles α^s and α^v are defined in figure 1. $E(\alpha^s)$ is the total radiation flux incident on the triangular facet with area $\mathcal{A}$. This flux is modeled as a sum of two terms. The first term corresponds to direct radiation with intensity $\mathcal{I}^s$ from the light source at infinity (commonly the sun). The second term corresponds to ambient light with intensity $\mathcal{I}^a$. The parameter θ in equation (12) is the angle between the camera axis and the viewing direction (the vector from the surface to the camera); κ is the lens falloff factor. $\Delta\Omega$ in (12) is the solid angle subtended by the camera which is determined by the area of the lens S and the distance d from the centroid of the triangular facet to the camera. If shadows are present on the surface the situation is somewhat more complex. In this paper we assume that there are no shadows or occlusions present in the images. However the presence of shadows and occlusions, whilst making more complex the computation of the image [7, 8] and its derivatives, should lead to a more precise and robust surface estimate, as *long-range correlations* are incorporated into the estimation.

The overall complexity of the rendering procedure and derivative calculation procedure scales as

$$C = \#\text{ images} \times \#\text{ triangles} \times \frac{\text{pixel area}}{\text{triangle area}}. \tag{13}$$

This can be seen from the algorithmic outline of the rendering step:

```
loop over images
 loop over surface triangles
  loop over affected pixels
   calculate fractions and derivatives
   calculate light contribution and
             derivatives
   pixelintensity += light * fractions
  end
 end
end
```

Where the fractions are those in equation 11. The corresponding derivatives are efficiently calculated as shown in the next section.

3 Efficient Derivative Computation

To compute the MAP estimates of $\{\mathbf{z}, \rho\}$ and Θ we must compute both the image $\hat{I}(\mathbf{z}, \rho, \Theta)$ and the derivative matrices $\mathbf{D}_\mathbf{z}$, $\mathbf{D}_\rho$ and $\mathbf{D}_\Theta$.

The derivatives with respect to the albedo values can easily be derived from equations 10 and 12. Note that because $0 < \rho < 1$, in practice, we work with transformed albedo values, where $\rho \to \log(\rho/(1-\rho))$.

Denoting by u the component of $\mathbf{z}$ or Θ that we are currently considering, the pixel intensity derivatives with respect to u have two components

$$\frac{\partial \hat{I}_p}{\partial u} = \sum_\triangle \left(f_\triangle^p \frac{\partial \Phi_\triangle}{\partial u} + \Phi_\triangle \frac{\partial f_\triangle^p}{\partial u} \right) \tag{14}$$

The first component is due to changes in angle – as the height of a vertex changes, the normal to the facet changes, and so the derivative has a component due to the change in angle between the normal and the sun direction; as the camera changes position, the angle between the normal and the ray to the camera changes.

Consider first $\partial \Phi_\triangle / \partial \Theta_i$. We neglect the derivatives with respect to the fall-off angle, θ, as their contribution will be small, and so it is clear from equation 12 that the derivative with respect to any of the camera orientation angles is zero.

The derivative with respect to the camera position parameters is given by

$$\begin{aligned} \frac{\partial \Phi_\triangle}{\partial \Theta_i} &\propto \frac{\partial}{\partial \Theta_i} \cos \alpha^v \\ &= \frac{\hat{\mathbf{n}}}{v}(\hat{z}_i - \hat{z}_v(\hat{z}_v.\hat{z}_i)) \end{aligned} \tag{15}$$

where $\mathbf{v}$ is the vector from the triangle to the camera, $v = |\mathbf{v}|$, Θ_i are the three components of the camera position, $\hat{z}_i$ are unit vectors in the three coordinate directions and $\hat{z}_v = \mathbf{v}/v$ (see figure 1). $\hat{\mathbf{n}}$ is the normal to the triangular facet.

Consider now the derivative with respect to the height of one of the mesh vertices, z_i. The flux derivative, $\partial\Phi/\partial z_i$, can be computed directly from the coordinates of the triangle vertices and the camera position using equation 12. For the surface triangle with vertices $(\mathbf{P}_{i_0}, \mathbf{P}_{i_1}, \mathbf{P}_{i_2})$ the flux derivative with respect to the z component of the vertex $\mathbf{P}_{i_0}$ equals

$$\frac{\partial\Phi}{\partial z_{i_0}} = \frac{1}{2}\rho(\mathbf{P}_{i_2} - \mathbf{P}_{i_1}) \times \hat{\mathbf{z}} \cdot \mathbf{g}\, \frac{S}{d^2}, \tag{16}$$

where

$$\mathbf{g} = \mathcal{I}_s(\hat{\mathbf{z}}_v \cos\alpha_s + \hat{\mathbf{z}}_s \cos\alpha_v - \hat{\mathbf{n}} \cos\alpha_s \cos\alpha_v) + \mathcal{I}_a \hat{\mathbf{z}}_v$$

and $\hat{\mathbf{z}}$ is a unit normal in the vertical direction.

For a triangle that projects entirely within a pixel, this completes the derivative computation – the second term in equation 14 is the derivative of the *fractional area* of the triangle that projects into the pixel.

3.1 Fractional Area Derivatives

When the height of a vertex, z, changes, its projection on the image plane, $\bar{\mathbf{P}}$, also moves, by $\delta\bar{\mathbf{P}}$. This gives rise to a change $\delta\bar{A}_\triangle$ in the area of the projection of the triangle, and also the change $\bar{A}_{\text{polygon}}$ in the polygon area. It follows from equation 11 that

$$\frac{\partial f^p_\triangle}{\partial z_{i_0}} = \frac{1}{\bar{A}_\triangle}\left(\frac{\partial\bar{A}_{\text{polygon}}}{\partial\bar{\mathbf{P}}_{i_0}} - f^p_\triangle \frac{\partial\bar{A}_\triangle}{\partial\bar{\mathbf{P}}_{i_0}}\right)\frac{\partial\bar{\mathbf{P}}_{i_0}}{\partial z_{i_0}}. \tag{17}$$

where the point displacement derivative $\partial\bar{\mathbf{P}}_{i_0}/\partial z_{i_0}$ can be found in [13].

However, when the camera parameters change, the positions of the projections of *all* the mesh vertices into the image plane will move. Then the the derivative of the fractional area is simply a sum of all three position changes and is given by

$$\frac{\partial f^p_\triangle}{\partial\Theta_i} = \frac{1}{\bar{A}_\triangle}\sum_{j=i_0,i_1,i_2}\left(\frac{\partial\bar{A}_{\text{polygon}}}{\partial\bar{\mathbf{P}}_j} - f^p_\triangle \frac{\partial\bar{A}_\triangle}{\partial\bar{\mathbf{P}}_j}\right)\frac{\partial\bar{\mathbf{P}}_j}{\partial\Theta_i}. \tag{18}$$

The point displacement derivatives are again in [13].

Thus, the task of computing the derivative of the area fraction given in equation 18 is reduced to the computation of $\partial\bar{A}_\triangle/\partial\bar{\mathbf{P}}_j$ and $\partial\bar{A}_{\text{polygon}}/\partial\bar{\mathbf{P}}_j$. Note that the intersection of a triangle and a pixel for a rectangular pixel boundary can, in general, be a polygon with 3 to 7 edges with various possible forms. However the algorithm for computing the polygon area derivatives that we have developed is general, and does not depend on a particular polygon configuration. The main idea of the algorithm can be described as follows. Consider, as an example, the polygon shown in figure 2 which is a part of the projected surface triangle with indices i_0, i_1, i_2. We are interested in the derivative of the polygon

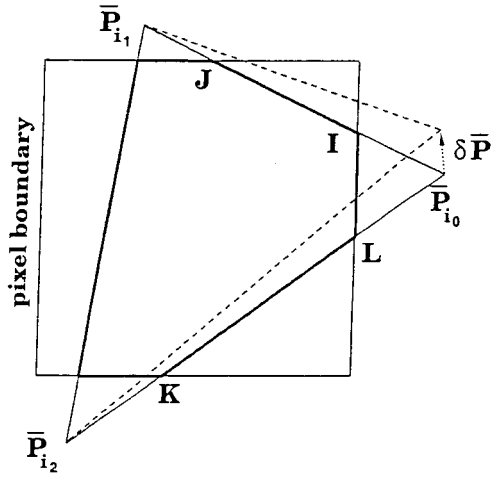

Fig. 2. The intersection of the projection of a triangular surface element (i_0, i_1, i_2) onto the pixel plane with the pixel boundaries. Bold lines corresponds to the edges of the polygon resulting from the intersection. Dashed lines correspond to the new positions of the triangle edges when point $\mathbf{P}_{i_0}$ is displaced by $\delta\mathbf{P}$

area with respect to the point $\bar{\mathbf{P}}_{i_0}$ that connects two edges of the projected triangle, $(\mathbf{P}_{i_2}, \mathbf{P}_{i_0})$ and $(\mathbf{P}_{i_0}, \mathbf{P}_{i_1})$. These triangular edges contain segments (**I**, **J**) and (**K**, **L**) that are sides of the corresponding polygon. It can be seen from figure 2 that when the point $\bar{\mathbf{P}}_{i_0}$ is displaced by $\delta\bar{\mathbf{P}}_{i_0}$ the change in the polygon area is given by the sum of two terms

$$\delta\bar{A}_{\text{polygon}} = \delta A_{\mathrm{I,J}} + \delta A_{\mathrm{K,L}}$$

These terms are equal to the areas spanned by the two corresponding segments taken with appropriate signs. Therefore the polygon area derivative with respect to the triangle vertex $\bar{\mathbf{P}}_{i_0}$ is represented as a sum of the two "segment area" derivatives for the two segments adjacent to a given vertex. Using straightforward geometrical arguments one can calculate the areas $\delta A_{\mathrm{I,J}}$ and $\delta A_{\mathrm{K,L}}$ to first order in the displacement $\delta\bar{\mathbf{P}}_{i_0}$. Then the polygon area derivative can be written in the following form:

$$\frac{\partial\bar{A}_{\text{polygon}}}{\partial\bar{\mathbf{P}}_{i_0}} = \frac{1}{2}\hat{\sigma}\cdot\mathbf{W}, \quad \hat{\sigma} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{19}$$

The unit antisymmetric matrix $\hat{\sigma}$ performs a $-\pi/2$ rotation in the image plane and vector **W** equals

$$\begin{aligned}\mathbf{W} = &\left[(1-R_{\mathrm{I}}^2) - R_{\mathrm{J}}^2\right](\bar{\mathbf{P}}_{i_0} - \bar{\mathbf{P}}_{i_2}) \\ &+ \left[(1-R_{\mathrm{K}}^2) - R_{\mathrm{L}}^2\right](\bar{\mathbf{P}}_{i_1} - \bar{\mathbf{P}}_{i_0}).\end{aligned} \tag{20}$$

The ratio factors R determine the positions of the intersection points $\mathbf{I}, \mathbf{J}, \mathbf{K}, \mathbf{L}$ on the edges of the triangle (see figure 2).
Equations 19 and 20 are the central result of the area fraction derivative computation. It is given for the general case of triangle-pixel intersection where two edges of triangle adjacent to the vertex $\mathbf{P}_{i_0}$ each have two intersection points. Note that pairs of intersection points, $\mathbf{I}, \mathbf{J}$ and $\mathbf{K}, \mathbf{L}$ are defined in a unique way if one considers the triangle edges in counterclockwise order. Therefore equations 19- 20 can be applied to all possible intersection cases. For example, assume that all three triangle vertices are projected inside the pixel. In this case intersection point $\mathbf{K}$ has merged with $\mathbf{P}_{i_2}$, points $\mathbf{L}$ and $\mathbf{I}$ have merged with $\mathbf{P}_{i_0}$ and $\mathbf{J}$ with $\mathbf{P}_{i_1}$. Then in equation 20 we should put

$$R_{\mathbf{K}} = R_{\mathbf{L}} = R_{\mathbf{I}} = R_{\mathbf{J}} = 0. \tag{21}$$

In this case polygon area derivative in equation 19 is reduced to the derivative of the full area of the projected triangle

$$\frac{\partial A'_{\triangle}}{\partial \bar{\mathbf{P}}_{i_0}} = \frac{1}{2}\hat{\sigma} \cdot (\mathbf{P}_{i_1} - \mathbf{P}_{i_2}) . \tag{22}$$

The general rule for computing the ratio factors $R_{\mathbf{I},\mathbf{J},\mathbf{K},\mathbf{L}}$ can be formulated as follows:

- If point $\mathbf{P}_{i_0}$ lies inside of the pixel one should set in equation 20 ratio factors $R_{\mathbf{L}} = 0$ and $R_{\mathbf{I}} = 0$.
- If point $\mathbf{P}_{i_2}$ lies inside of the pixel then one sets $R_{\mathbf{K}} = 0$.
- If $\mathbf{P}_{i_1}$ lies inside then $R_{\mathbf{J}} = 0$.

This describes all possible intersection cases and provides a full description for the area fraction derivative (18).

Further details of the derivative computation, together with full details of the point displacement derivatives, can be found in [13].

4 Results

We present here the results of applying our methodology. We will demonstrate our contention that the *small improvements* made by our registration method to the camera parameter estimates results in *large improvements* to the quality of the inferred surface.

Figure 3 shows four synthetic images of a region of Duckwater, Nevada. They were generated by rendering a synthetic surface. The surface was constructed by using the USGS Digital Elevation Model for the heights, and using the scaled intensities of a LANDSAT-TM image as surrogate albedos. The size of the surface is 301×301 points. The distance between grid points was taken to be one unit, and the heights scaled appropriately. Figure 4 shows a perspective view of the surface with expanded vertical scale. Table 1 gives the camera parameters that were used to generate the images.

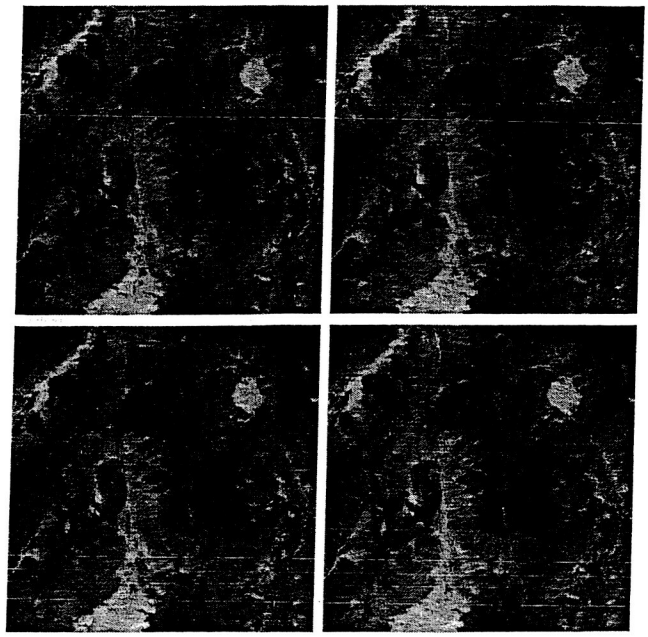

Fig. 3. Four synthetic images of Duckwater, Nevada

An initial estimate of the camera parameters was made by using point matching [12]. We have found that the Harris corner detector [11] typically used to select features does not find many reliable features in the types of natural scenery we are concerned with here. Table 1 gives the parameters estimated by matching five points across the four images. Note that these camera parameter estimates appear accurate, with the major error being in the orientation angle (view-up vector).

Using these estimated camera parameters, a dense surface estimate can be made. For space reasons we do not show the surface estimate, instead, in figure 7 we show the error surface, and a cross section in figure 6. The main points to note are that

1. the small inaccuracies in the camera parameter estimation have resulted in an erroneous slope in the surface estimate.
2. the overall height of the surface is shifted upwards; but note that the overall shift is a small percentage (less than 0.5%) of the distance from the surface to the cameras. The overall height is only weakly determined.
3. the albedo estimates are in general quite good (the RMSE for the albedo estimate is 0.022).

Using the gradient-based, whole image, approach to camera calibration to a surface, that we have described above, we then registered the images to the surface estimate. Using the new camera parameters, we re-estimated the surface.

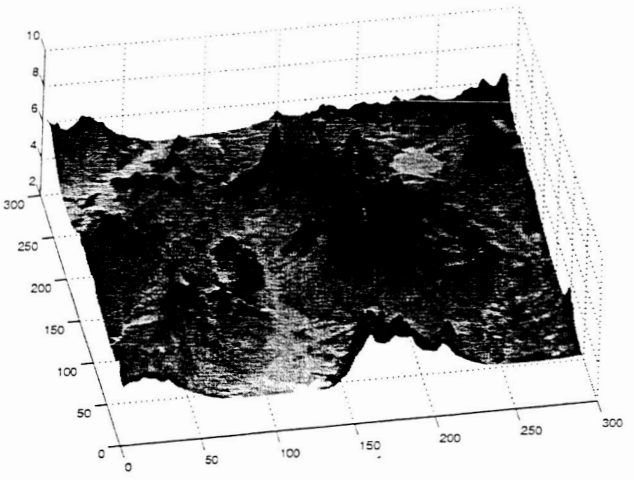

Fig. 4. True surface

This was iterated three times. On a 1.2GHz Athlon PC, rendering and computing the derivative matrix takes less than 2 seconds per image. Convergence of the Conjugate Gradient for updating the surface estimate is achieved in around 200 seconds, and for updating the camera parameters in less than a second. Table 1 gives the final camera parameters, and figure 5 shows the final surface estimate. Again, note that the improvements to the registration parameters are small, but figures 8 and 6 show that these small improvements are crucial. Figure 8 is the error surface and figure 6 is a section through the error surfaces. We note the following:

1. the main improvement in the camera parameter estimation is in the orientation angle, defined by the view-up vector
2. the erroneous slope has been corrected
3. the error in the global height remains
4. the estimate shows most inaccuracies close to rapid changes in albedo, for example the white (salt lake) area to the top right of the surface, where albedo and slope effects have not been completely decoupled.

From these numerical experiements, it is clear that the quality of the surface inference is very sensitive to even small changes in the camera parameters. The *convergence radius* of a successful surface reconstruction with respect to the camera parameters is quite small, and therefore the improvements our registration method give, whilst appearing to be small, have a large effect on the accuracy of the surface estimate.

5 Conclusions

In this paper we have described a system that takes a set of images and uses them to infer both the *camera parameters* and a *dense surface model.* It does this by iterative linearization of a model of the image formation process, and minimization of the error between the whole of the observed and rendered images with respect to the camera and surface parameters. We have demonstrated the convergence of this system on a set of images rendered from a model of a region of Nevada. We have demonstrated the need for *extremely accurate* camera registrations in order to accurately infer a dense surface model, and have shown that our registration method achieves this.

Though the computational cost of our system is high compared to a conventional 3D reconstruction algorithm, it is still of linear complexity, and he system we have described has many advantages. The accurate, dense surface reconstruction which also has albedo information can be used for a number of scientific applications, for example spectroscopy for remote mineral type determination. The scale of the surface model that is estimated is decoupled from the pixel scale of the images via the rendering process. This means that the surface model scale can be chosen by the user, either on the basis of the use to which the surface model will be put, or a scale may be chosen which is best justified by the image data. This is important – if we have many low resolution images of a region, the scale of the surface model may be super-resolved (where a triangular surface element projects onto an area smaller than a pixel on the image plane). If the coverage of the surface by the images is non-uniform, we can specify a spatially-varying mesh for the surface, denser in regions where we have more images.

The information about the surface captured by the system is not just the MAP surface estimate, but also the accuracy of the estimate, represented by the inverse covariance matrix ($\mathbf{A}$ in equation 5). Knowing the inverse covariance matrix allows for recursive updates – as new images become available the information they contain can be integrated into the model. In Bayesian terminology, the posterior distribution from one set of images (defined by the MAP estimate and the inverse covariance matrix) becomes the prior for estimation with new images.

Finally, we are not restricted to only image data. If data from other sensing modalities is available (for example, laser altimetry data) then we can add a term to the likelihood (equation 2) for this data, take derivatives of a model of how this new sensor makes measurements with respect to the surface model parameters, and our surface model estimate will seamlessly integrate the multimodal information.

References

1. J. Foley, A. van Dam, S. Finer, and J. Hughes. *Computer Graphics, principles and practice.* Addison-Wesley, 2nd ed. edition, 1990.

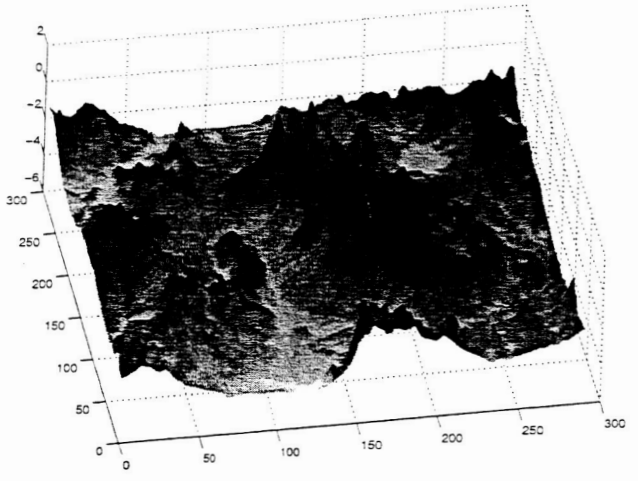

Fig. 5. Inferred surface

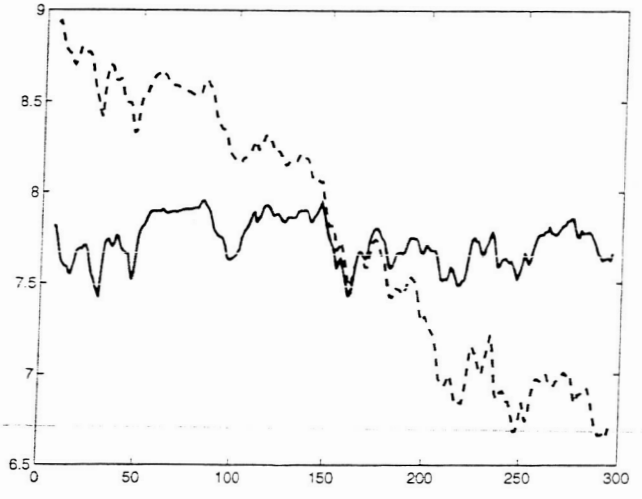

Fig. 6. Section through the error surfaces. Dotted line - section through the error surface from the pointmaching surface estimate; solid line - section through the error surface of the final estimate

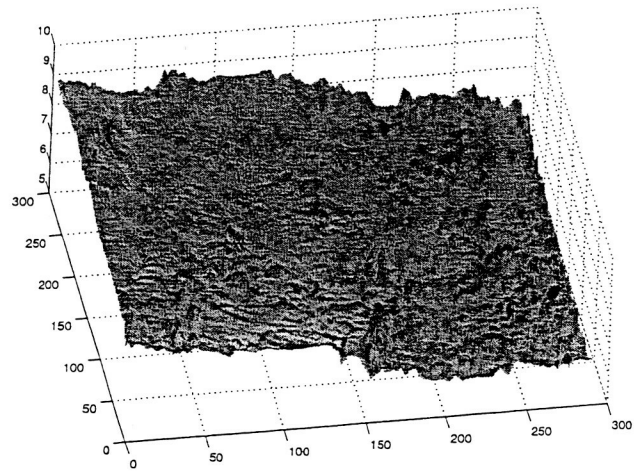

Fig. 7. Error surface for the surface estimate using camera parameters from point matching. The shape of the surface gives the error in the heights, and the colouring of the surface is propotional to the error in the albedo.

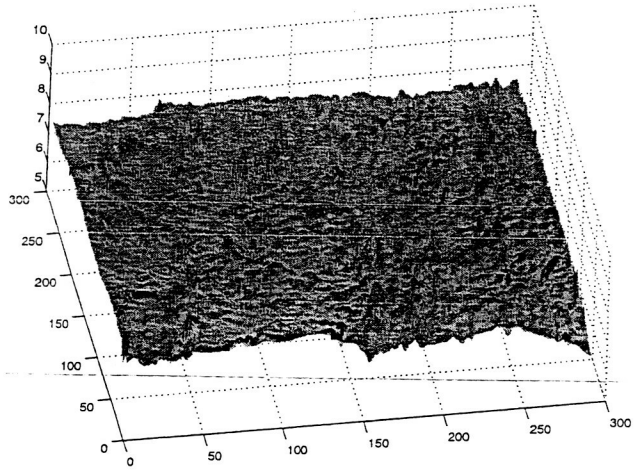

Fig. 8. Error surface for the surface estimate using iteratively refined camera parameters

		True	Pointmatch	Gradient-based
image 1	camera	$(-100, 150, 1700)$	$(-100, 150, 1700)$	$(-100.0, 150.0, 1700.0)$
	look at	$(150, 150, 0)$	$(150, 150, 0)$	$(150.0, 150.0, 0)$
	view up	$(0, 1, 0)$	$(0, 1, 0)$	$(-0.00084, 1, 9.0 \times 10^{-5})$
image 2	camera	$(300, 150, 1700)$	$(301, 150, 1700)$	$(301.0, 150.0, 1700.0)$
	look at	$(150, 150, 0)$	$(152, 150, 0)$	$(151.9, 150.1, 0)$
	view up	$(0, 1, 0)$	$(0.0031, 1, -0.00031)$	$(0.00066, 1, -2.38 \times 10^{-5})$
image 3	camera	$(150, -100, 1700)$	$(151, -101, 1700)$	$(151.0, -101.0, 1700.0)$
	look at	$(150, 150, 0)$	$(151, 149, 0)$	$(151.1, 149.0, 0)$
	view up	$(0, 1, 0)$	$(-0.0032, 0.989, 0.146)$	$(0.00039, 0.989, 0.146)$
image 4	camera	$(150, 345, 1700)$	$(153, 348, 1700)$	$(153.0, 348.0, 1700.0)$
	look at	$(150, 150, 0)$	$(151, 151, 0)$	$(151.0, 151.0, 0)$
	view up	$(0, 1, 0)$	$(0.0021, 0.933, -0.116)$	$(0.00055, 0.9934, -0.115)$

Table 1. Camera parameters used to generate the images in figure 3 ("True"), estimated using point matching ("Pointmatch") (The first image was taken as a known reference.) and final camera parameters estimated using gradient based image error estimation on the estimated surface (3rd iteration)("Gradient-based").

2. A.L. Yuille, D. Snow, R. Epstein and P.N. Belhumeur Determining Generative Models of Objects Under Varying Illumination: Shape and Albedo from Multiple Images Using SVD and Integrability. International Journal of Computer Vision, 35(3), 203-222, 1999.
3. P. Fua and Y.G. Leclerc Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading. International Journal of Computer Vision, September 1995.
4. P. Fua and Y.G. Leclerc Registration Without Correspondences. International Conference on Computer Vision and Pattern Recognition, Seattle, WA, pp. 121-128, Jun 1994.
5. V.N. Smelyanskiy, P. Cheeseman, D.A. Maluf and R.D. Morris Bayesian Super-Resolved Surface Reconstruction from Images. Proceedings of International Conference on Computer Vision and Pattern Recognition, June 2000
6. J. Bernardo and A. Smith. *Bayesian Theory.* Wiley, Chichester, New York, 1994.
7. K. Weiler and P. Atherton. *Hidden Surface Removal Using Polygon Area Sorting* Proceedings of SIGGRAPH, pp 214-222, 1977.
8. E. Catmull *A Hidden-Surface Algorithm with Anti-Aliasing* Proceedings of SIGGRAPH, pp 6-11, 1978.
9. O. Faugeras. *Three-Dimensional Computer Vision.* MIT Press, 1993.
10. Z. Zhang. A Flexible New Technique for Camera Calibration. Technical Report MSR-TR-98-71, Microsoft Research, Redmond, Washington.
11. C. Harris. A Combined Corner and Edge Detector. Proceedings of the Alvey Vision Conference, pp 189-192, 1987.
12. Z. Zhang, R. Deriche, O. Faugeras,and Q.T. Luong. A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry. AI Journal, vol. 78, pp 87-119, 1994.
13. V.N. Smelyanskiy, R.D. Morris, D.A. Maluf and P. Cheeseman, (Almost) Featureless Stereo – Calibration and Dense 3D Reconstruction Using Whole Image Operations. Technical Report TR01-26, 2001, RIACS. www.riacs.edu